Deep Learning and Inverse Problems in Geoscience

R. Fablet et al.
ronan.fablet@imt-atlantique.fr
web: cia-oceanix.github.io

Introduction to Deep Learning

Inverse Problems in Geoscience

Mathematical formulation for inverse Problems

Inverse problems & Deep learning

Applications to geophysical dynamics

Inverse Problems in Geoscience

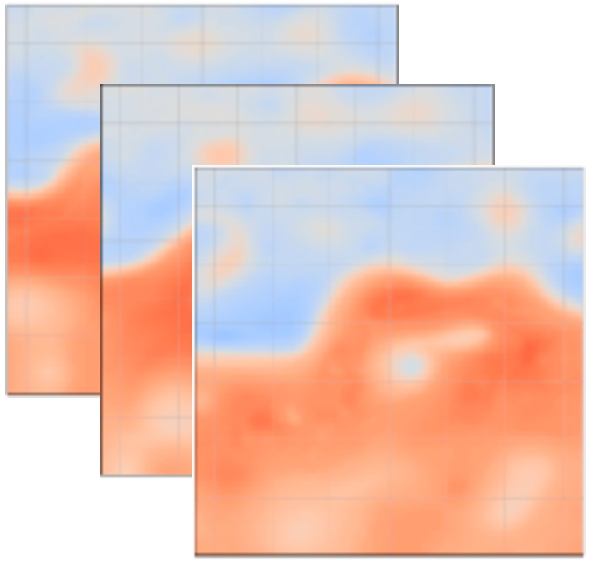
Mathematical formulations for inverse Problems

Inverse problems as learning problems

Applications to geophysical dynamics

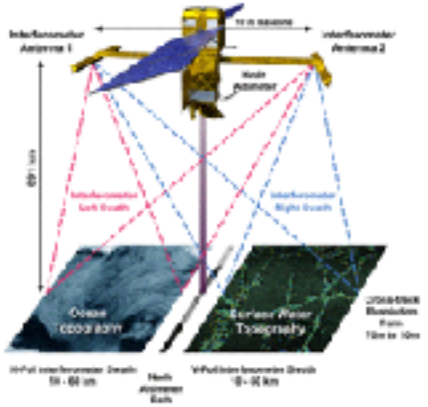
Inverse Problems in Geoscience: Generic formulation

State



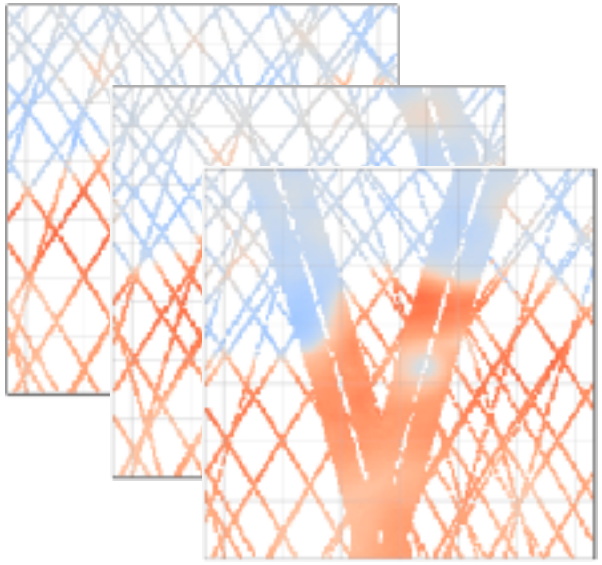
X

Observing system



Forward model H()

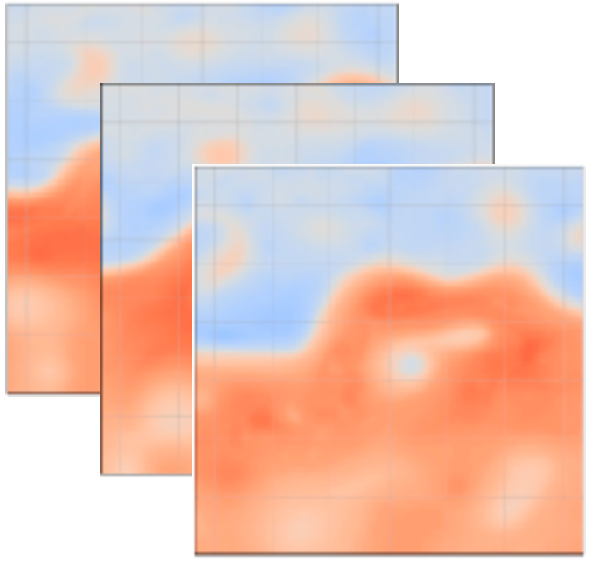
Observation



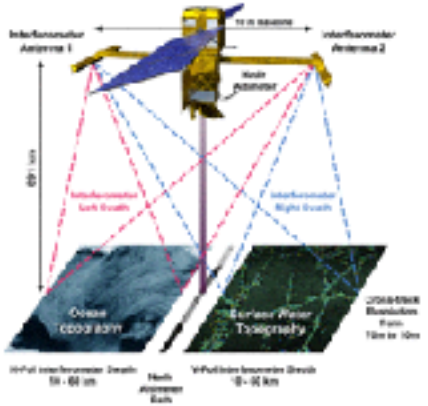
Y

Inverse Problems and ill-posedness

State

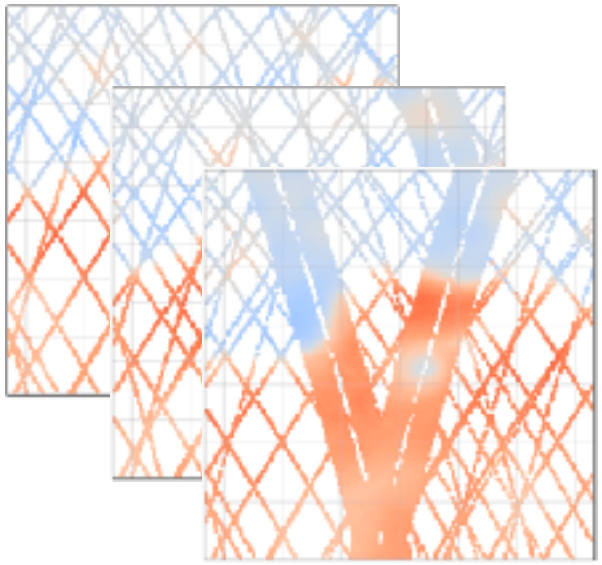


Observing system



Forward model H()

Observation



X

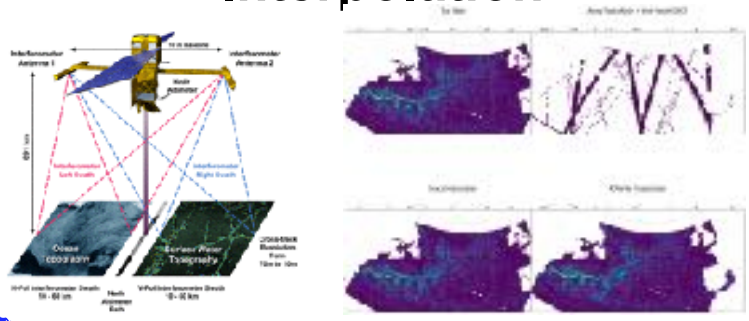
Y

Inverse model $H^{-1}()$

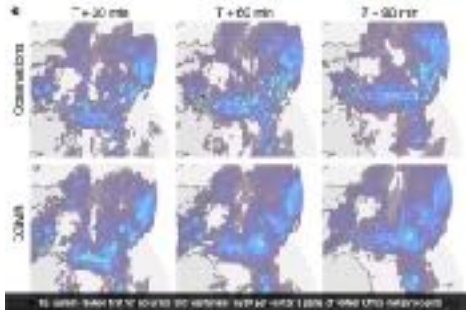
We need some additional information (prior) to make the problem invertible

Inverse Problems in Geoscience: some examples

Interpolation

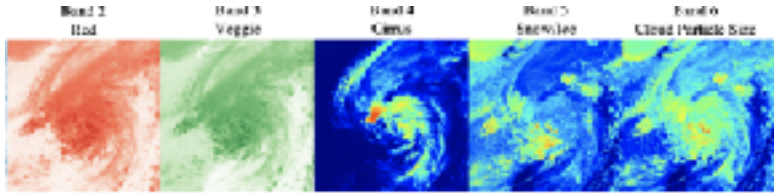


Obs.-driven Forecasting



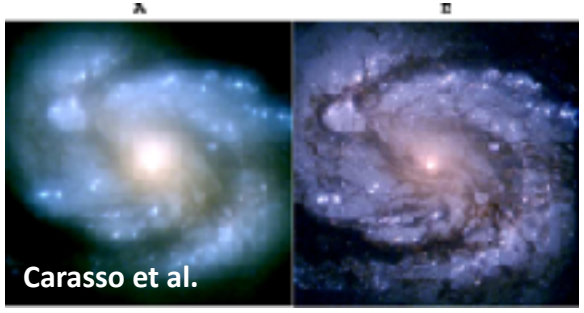
Deepmind

Multimodal fusion



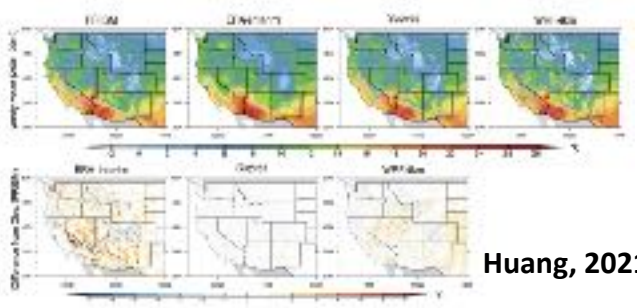
Vandal et al.

Deconvolution



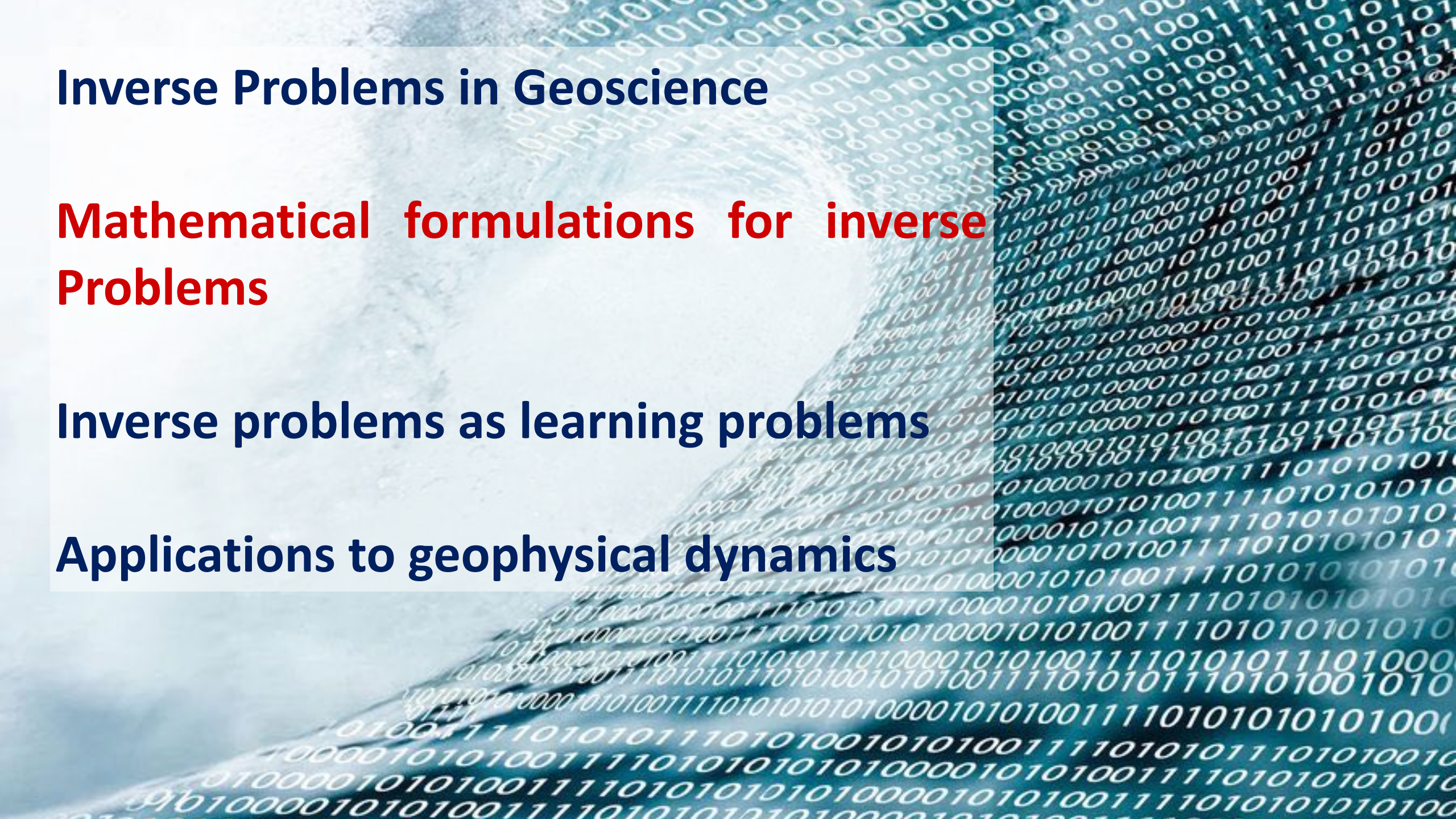
Carasso et al.

Downscaling

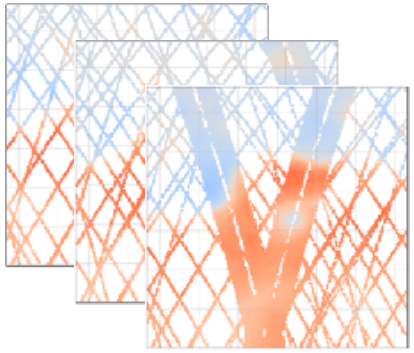


Huang, 2021

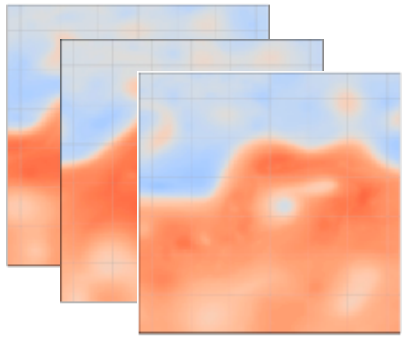
....


Inverse Problems in Geoscience
Mathematical formulations for inverse Problems
Inverse problems as learning problems
Applications to geophysical dynamics

(Variational) Data Assimilation: (Weak-Constraint) 4DVar DA



Partial observations y



True states x

State-space formulation:

$$\begin{cases} \dfrac{\partial x(t)}{\partial t} &= \mathcal{M}\left(x(t)\right) + \eta(t) \\ \\ y(t,p) &= x(t,p) \quad \forall t, \forall p \in \Omega_t \end{cases}$$

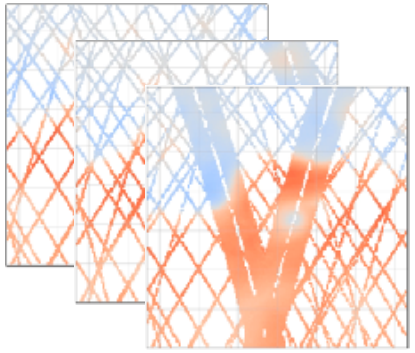
Associated variational formulation:

$$\arg\min_x \lambda_1 \sum_i \|x(t_i) - y(t_i)\|^2_{\Omega_{t_i}} + \lambda_2 \sum_n \|x(t_i) - \Phi(x)(t_i)\|^2$$

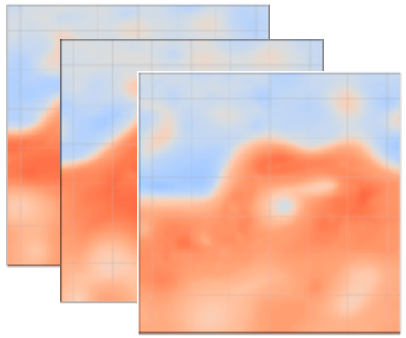
with $$\Phi(x)(t) = x(t-\Delta) + \int_{t-\Delta}^{t} \mathcal{M}\left(x(u)\right) du$$

$$\arg\min_x \lambda_1 \|x - y\|^2_\Omega + \lambda_2 \|x - \Phi(x)\|^2$$

Inverse problems stated as minimisation problems



Partial observations y



True states x

Minimization problem

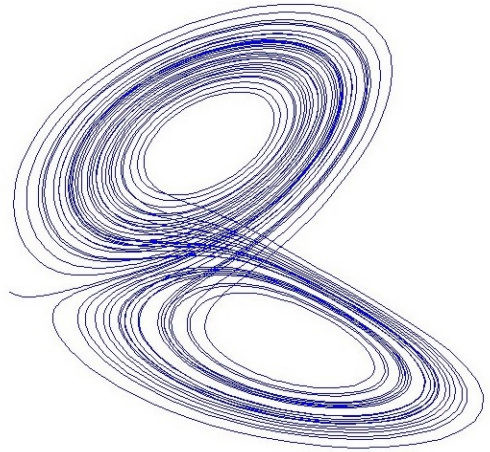
$$\widehat{\mathbf{x}} = \arg\min_{\mathbf{x}} \underbrace{\|\mathbf{y}-\mathbf{x}\|_{\Omega}^{2} + \lambda \|\mathbf{x}-\Phi(\mathbf{x})\|^{2}}_{U(\mathbf{x},\mathbf{y})}$$

How to solve the minimization ?

$$\begin{cases} X^{(0)} & = & X_0 \\ X^{(k+1)} & = & X^{(k)} - \alpha \nabla_X U\left(X^{(k)}, Y\right) \end{cases}$$

Can we use Pytorch to implement the minimization ?

(Variational) Data Assimilation: (Weak-Constraint) 4DVar DA



Minimization problem

$$\arg\min_x \lambda_1 \|x - y\|_\Omega^2 + \lambda_2 \|x - \Phi(x)\|^2$$

with $$\Phi(x)(t) = x(t - \Delta) + \int_{t-\Delta}^{t} \mathcal{M}(x(u))\, du$$

$$\begin{aligned}
\frac{\mathrm{d}x(t)}{\mathrm{d}t} &= \sigma\,(y(t) - x(t)) \\
\frac{\mathrm{d}y(t)}{\mathrm{d}t} &= x(t)\,(\rho - z(t)) \; - y(t) \\
\frac{\mathrm{d}z(t)}{\mathrm{d}t} &= x(t)\,y(t) - \beta\,z(t)
\end{aligned}$$

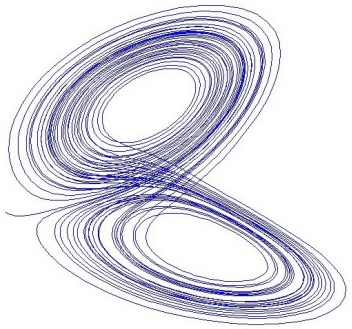
Lorenz-63 equations

(Variational) Data Assimilation: (Weak-Constraint) 4DVar DA

Numerical integration scheme as residual neural networks (eg, Fablet et al., 2018)

$$\begin{aligned}\frac{\mathrm{d}x(t)}{\mathrm{d}t} &= \sigma\left(y(t)-x(t)\right)\\ \frac{\mathrm{d}y(t)}{\mathrm{d}t} &= x(t)\left(\rho-z(t)\right)-y(t)\\ \frac{\mathrm{d}z(t)}{\mathrm{d}t} &= x(t)\,y(t)-\beta\,z(t)\end{aligned}$$

Lorenz-63 equations

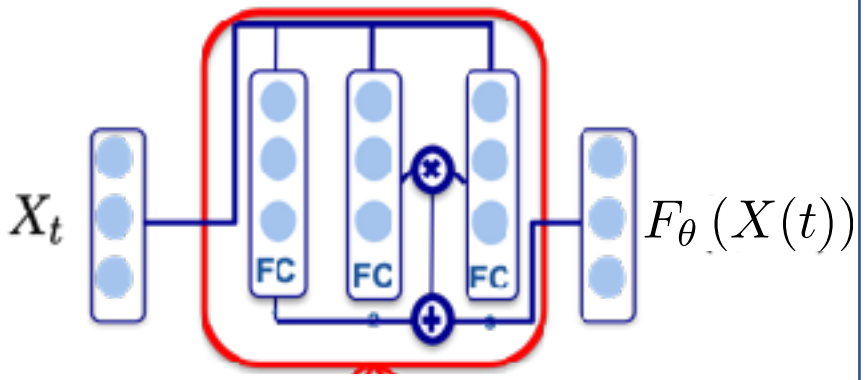


Associated Euler integration scheme

$$d_t X(t) = F_\theta\left(X(t)\right)$$

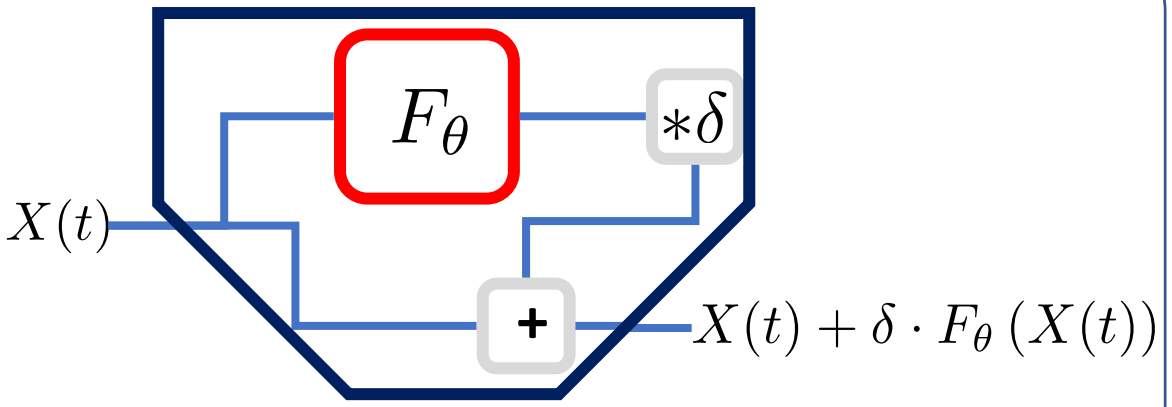
$$X(t+\delta) = X(t) + \delta \cdot F_\theta\left(X(t)\right)$$

NN architecture for differential operator



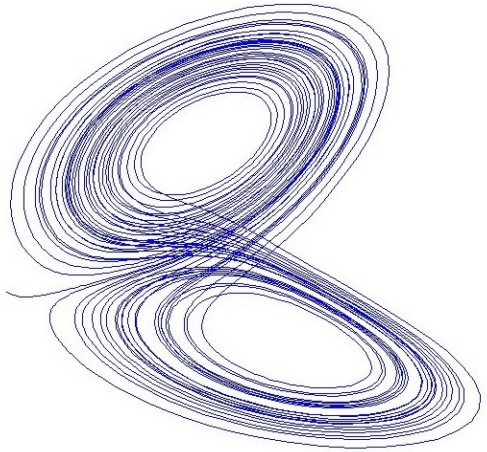
Bilinear architecture

NN architecture for integration scheme



ResNet architecture (Residual Network)

(Variational) Data Assimilation: (Weak-Constraint) 4DVar DA



Minimization problem

$$\arg\min_x \lambda_1 \|x-y\|_\Omega^2 + \lambda_2 \|x-\Phi(x)\|^2$$

with $$\Phi(x)(t) = x(t-\Delta) + \int_{t-\Delta}^{t} \mathcal{M}(x(u))\,du$$

$$\begin{aligned}
\frac{\mathrm{d}x(t)}{\mathrm{d}t} &= \sigma\,(y(t)-x(t)) \\
\frac{\mathrm{d}y(t)}{\mathrm{d}t} &= x(t)\,(\rho - z(t)) - y(t) \\
\frac{\mathrm{d}z(t)}{\mathrm{d}t} &= x(t)\,y(t) - \beta\,z(t)
\end{aligned}$$

Lorenz-63 equations

Let's try to implement this minimisation with Pytorch.

https://github.com/CIA-Oceanix/DLCourse_MOi_2022/blob/main/notebooks/notebook_Pytorch_Weak4DVar_L63.ipynb

Key message for (variational) Model-based Methods

- ***Solution stated as the minimisation of a variational cost***
- ***Explicit knowledge of observation model and prior***
- ***Implementation of associated minimisation algorithm***
- ***Analytical derivation of gradient operator (***e.g., Euler-Lagrange equations, Adjoint Methods***)***
- ***Implementation using DL schemes (***eg, Pytorch***) do not require the analytical gradient operator***

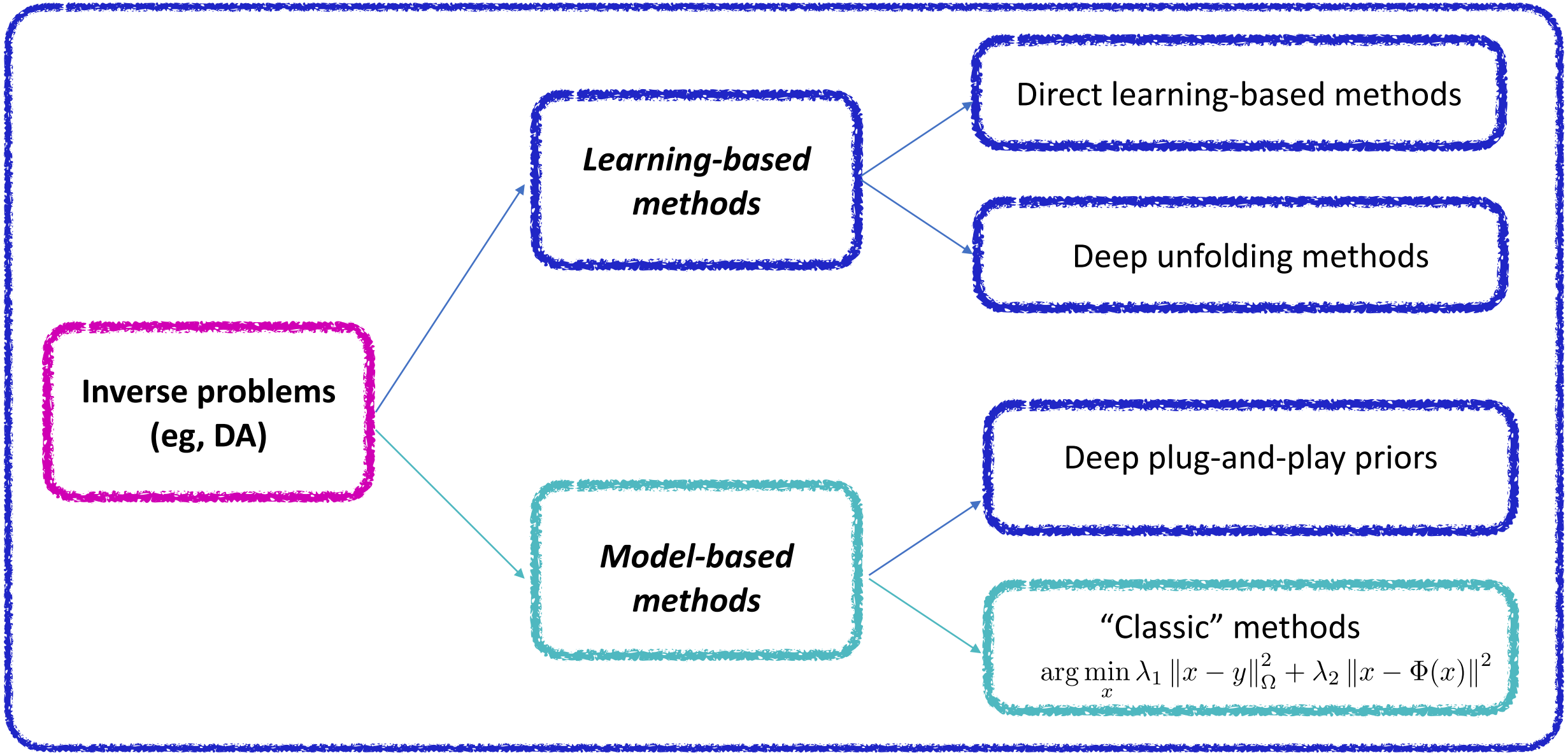
Inverse Problems in Geoscience

Mathematical formulations for inverse Problems

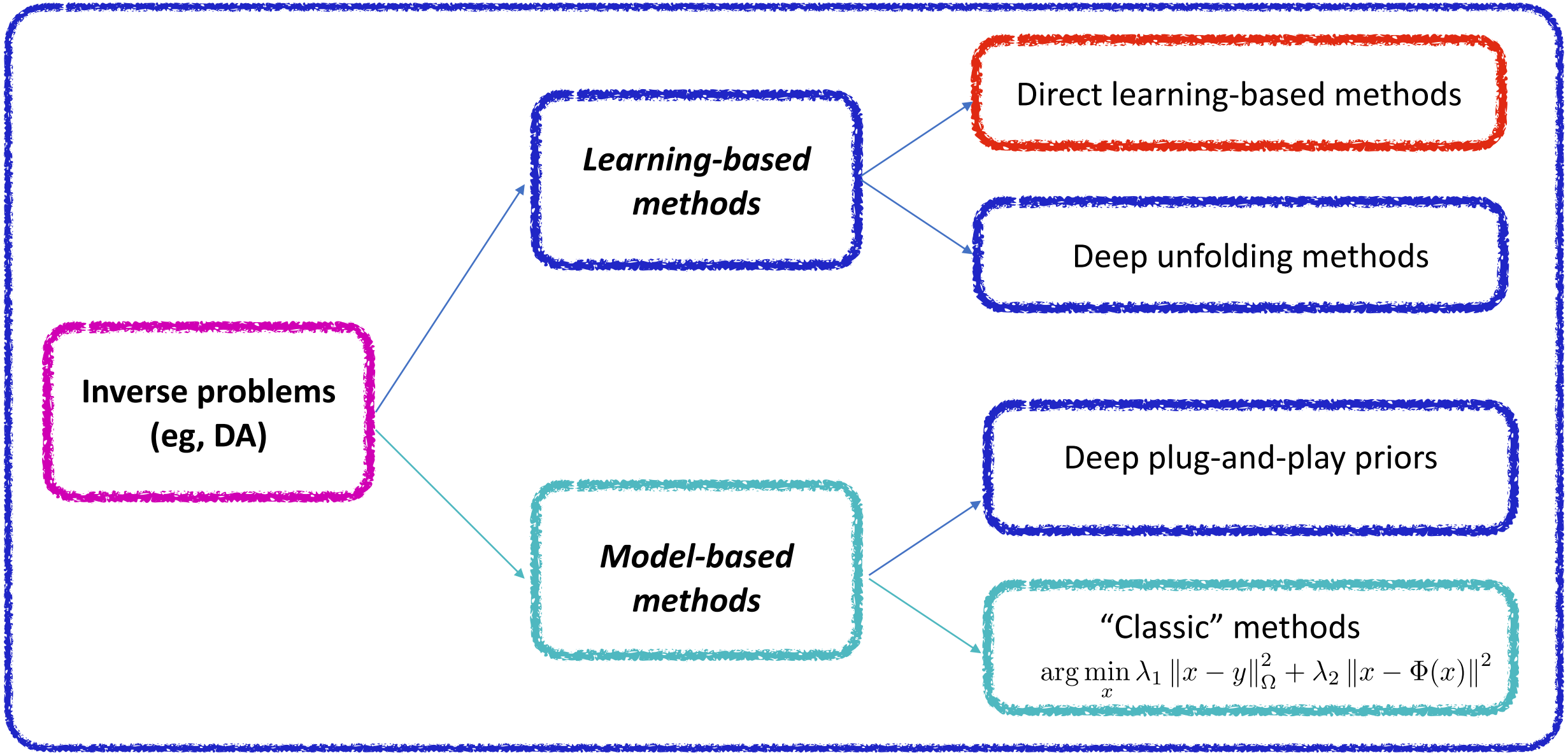
Inverse problems & Deep learning

Applications to geophysical dynamics

Model-driven vs. Learning-based approaches

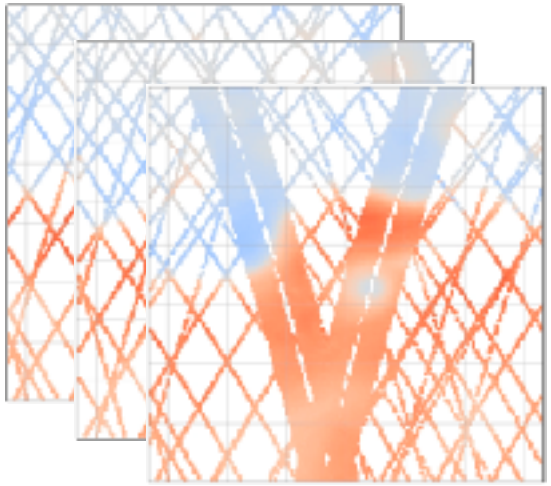


Model-driven vs. Learning-based approaches



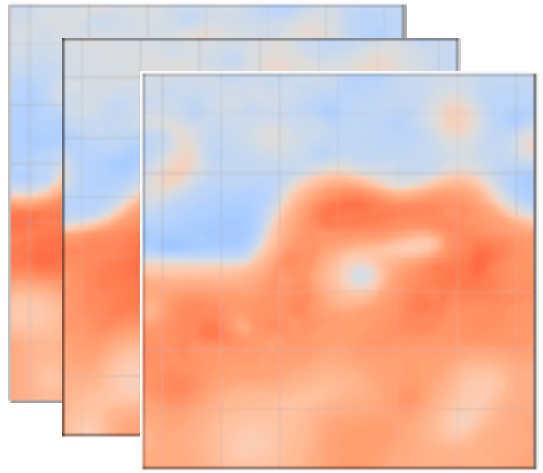
End-to-end learning for inverse problems

End-to-end architecture



Partial observations y

Deep learning Models



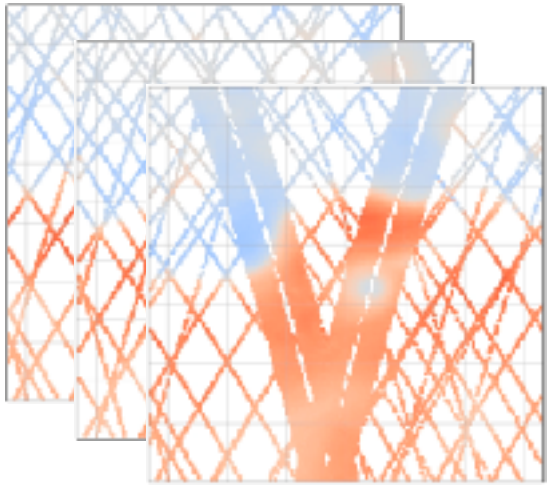
True states x

Which training loss ?

Which models / architectures ?

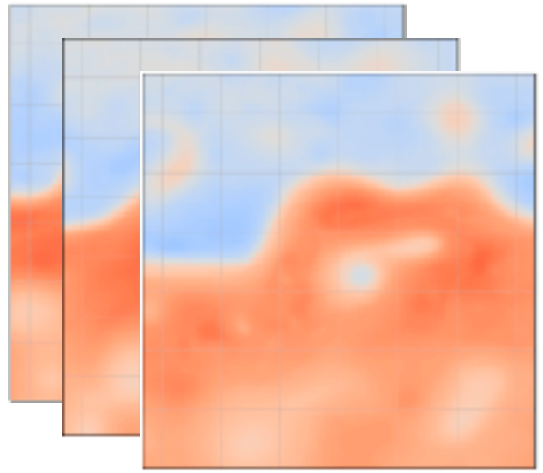
End-to-end learning for inverse problems

End-to-end architecture



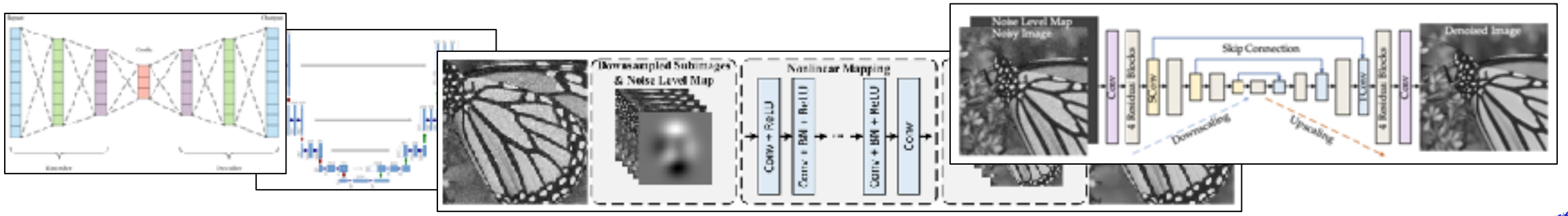
Partial observations y

Deep learning Models

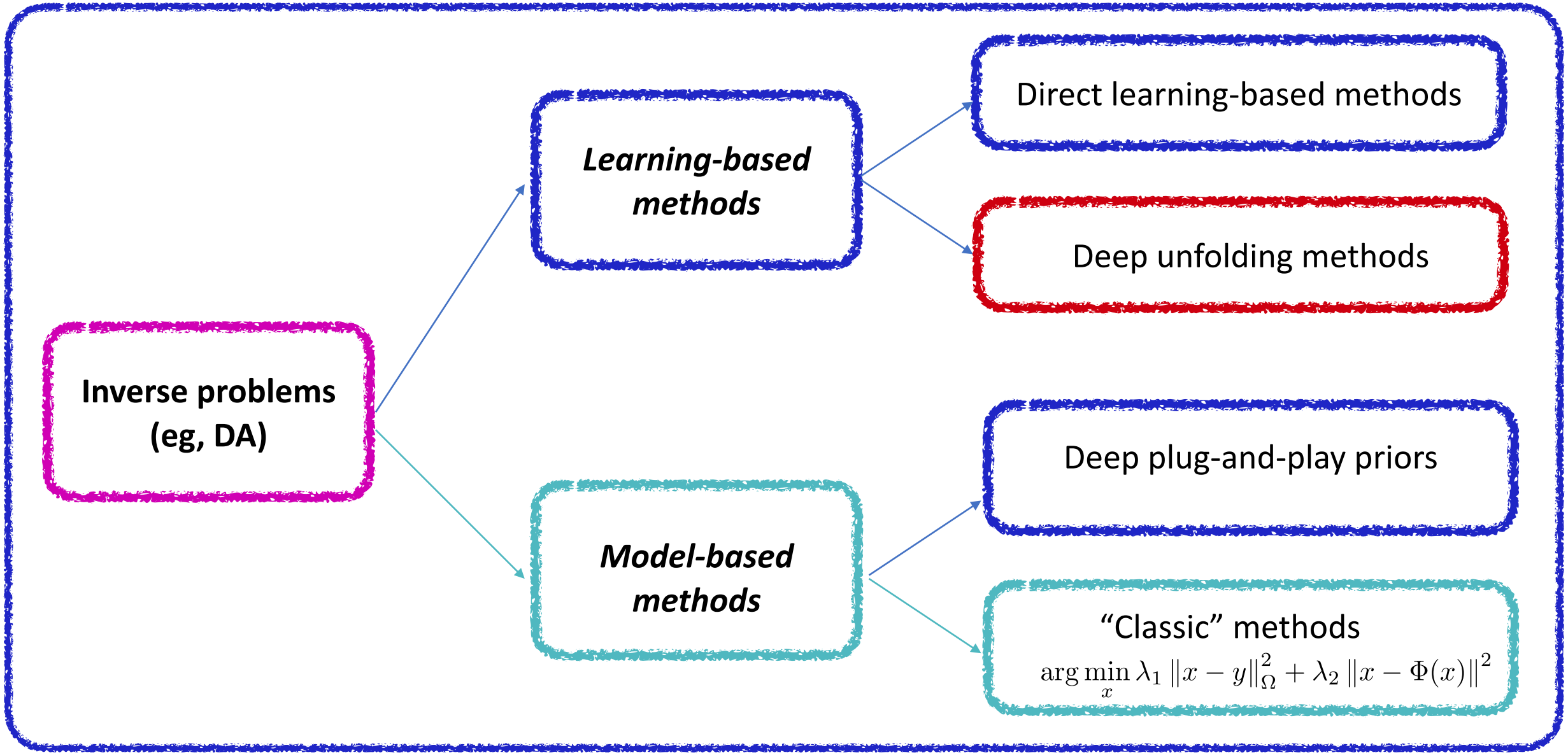


True states x

Which architectures? State-of-the-art CNN architectures?



Model-driven vs. Learning-based approaches

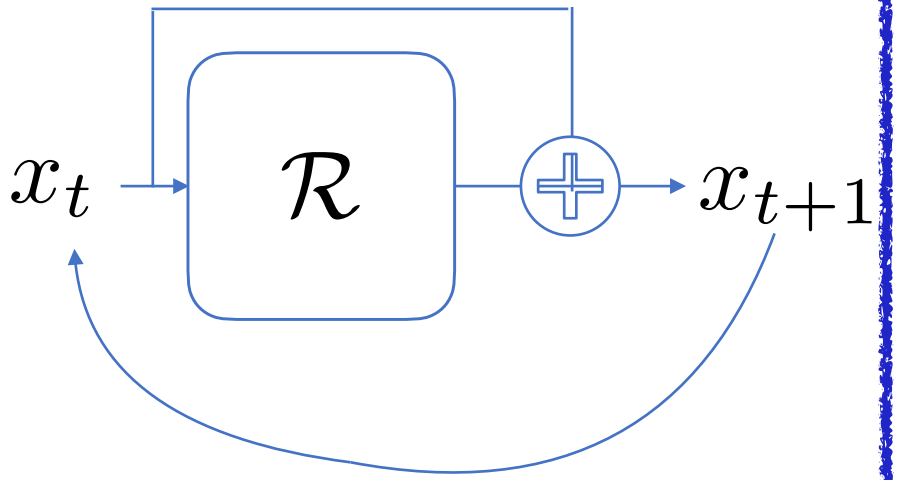


Folded vs. Unfolded Representations

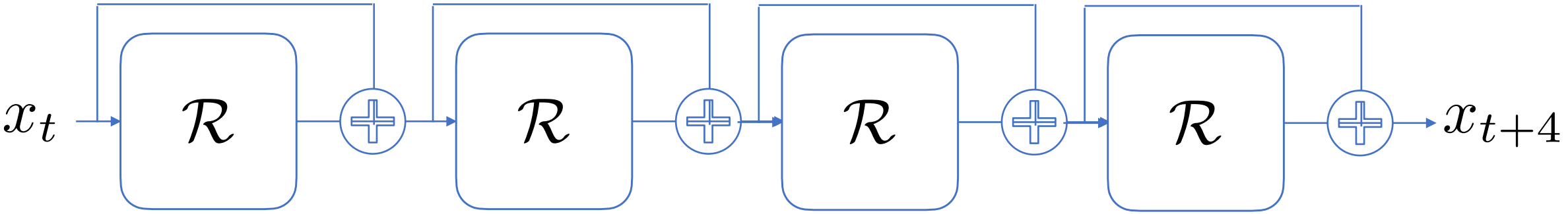
An example with a ResNet

$$x_{t+1} = x_t + \mathcal{R}(x_t)$$

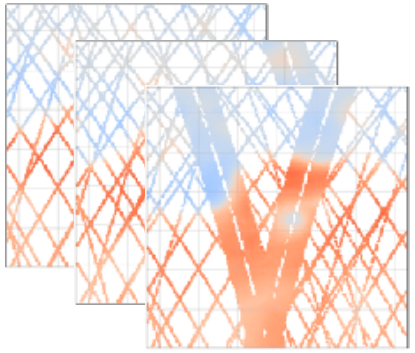
Folded Representation



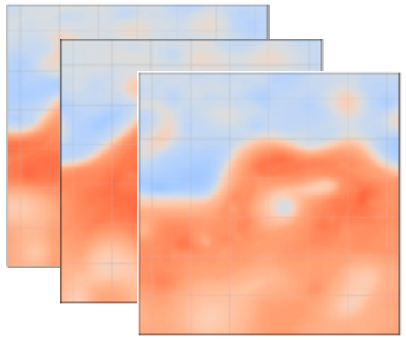
Unfolded Representation



Inverse problems using Deep unfolding schemes



Partial observations y



True states x

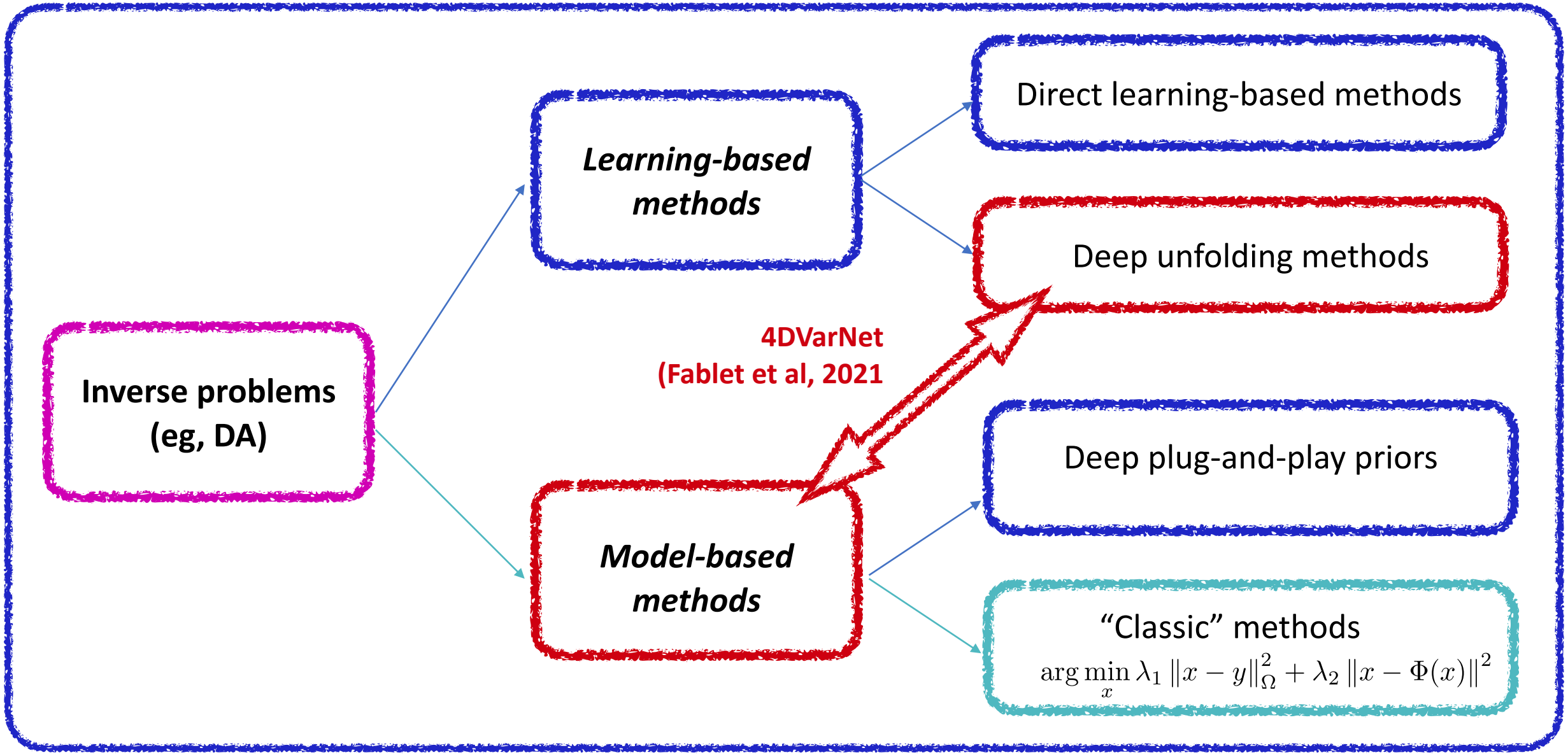
Basic idea: exploit knowledge on optimisation algorithms for inverse problems

- **Many schemes involve iterative algorithms**
- **On may unfold an Iterative procedure to define a deep learning architecture**

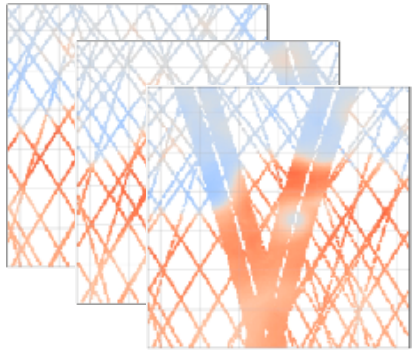
Examples

- **Image denoising/Unfolding of reaction-diffusion schemes** (e.g., Chen et al., 2015)
- **Medical imaging/Unfolding of ADMM schemes** (e.g., Yang et al., 2016)
- **Interpolation/Unfolding of fixed-point algorithms** (e.g., Fablet et al., 2020)
- **DA/Deep unfolding of sequential DA algorithms** (e.g., Boudier et al., 2020)

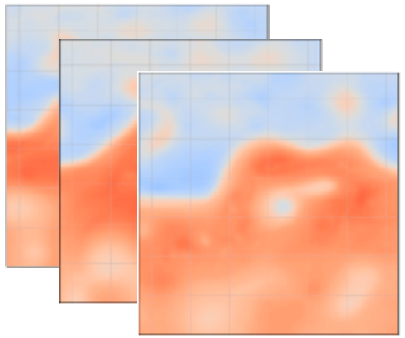
Model-driven vs. Learning-based approaches



Inverse problems: Unfolding 4DVar DA



Partial observations y



True states x

Gradient-based solver (adjoint/Euler-Lagrange method):

$$\arg\min_x \underbrace{\lambda_1 \|x-y\|_\Omega^2 + \lambda_2 \|x-\Phi(x)\|^2}_{U_\Phi(x,y,\Omega)}$$

Iterative update:

$$x^{(k+1)} = x^{(k)} - \alpha \nabla_x U_\Phi \left(x^{(k)}, y, \Omega \right)$$

Associated residual architecture ?

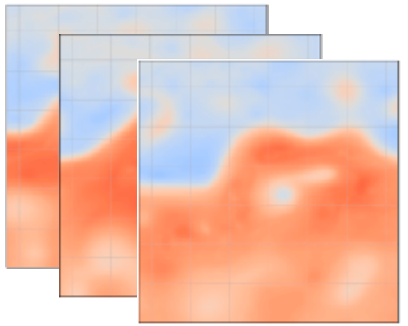
Inverse problems: Unfolding 4DVar DA

Iterative update: $x^{(k+1)} = x^{(k)} - \alpha \nabla_x U_\Phi \left(x^{(k)}, y, \Omega \right)$

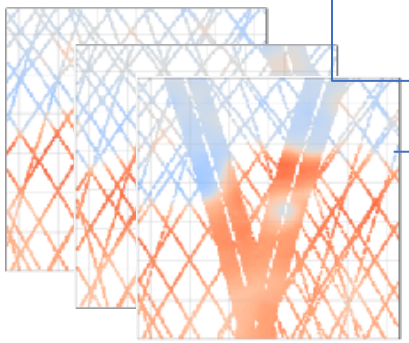
Initial state $x^{(0)}$

ResNet architecture

RU $x^{(1)}$ (....) RU $x^{(K)}$ RU $\tilde{x}$



Reconstructed states x



Observation data y, Ω

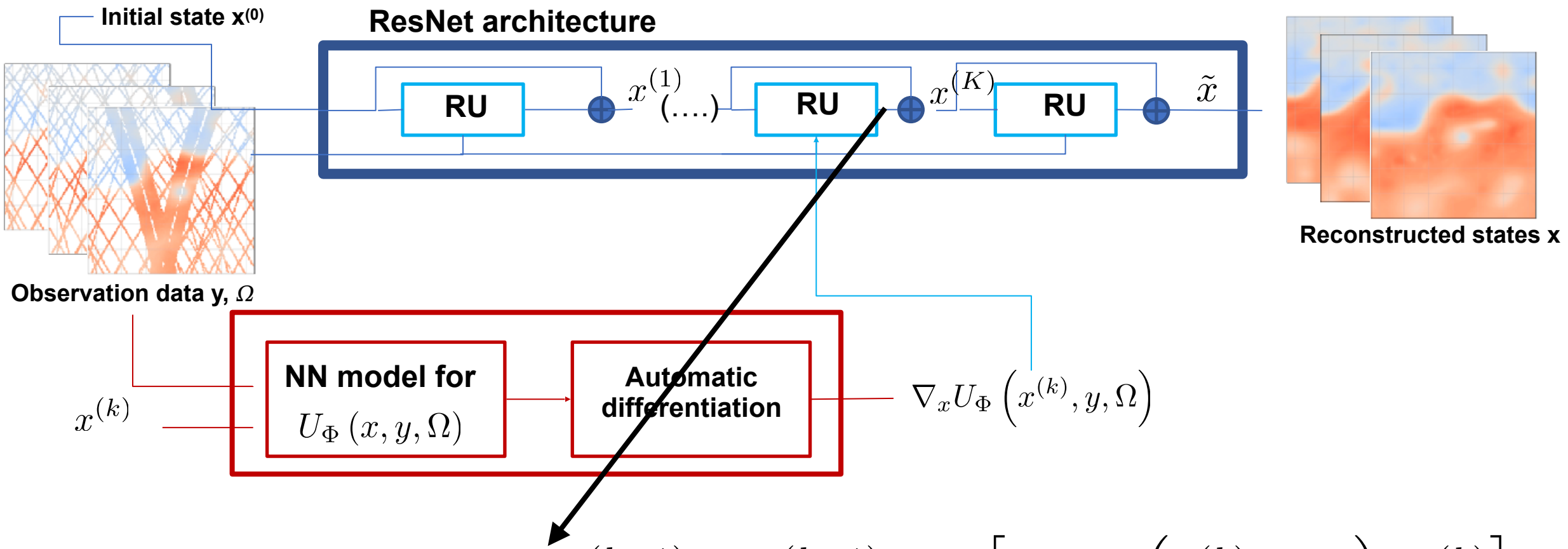
$x^{(k)}$

NN model for $U_\Phi(x, y, \Omega)$

Automatic differentiation

$\nabla_x U_\Phi \left(x^{(k)}, y, \Omega \right)$

Inverse problems: Unfolding 4DVar DA and Meta-learning



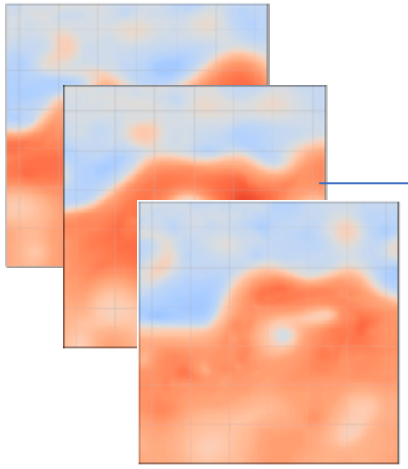
Generalised trainable update: $x^{(k+1)} = x^{(k+1)} + \mathcal{L}\left[\nabla_x U_\Phi\left(x^{(k)}, y, \Omega\right), z^{(k)}\right]$

Similar to learning-to-learn idea (*Andrychowicz et al., 2016*) (cf. Lecture #2, L. Drumetz)

End-to-end learning for 4DVar DA: projection operator Φ

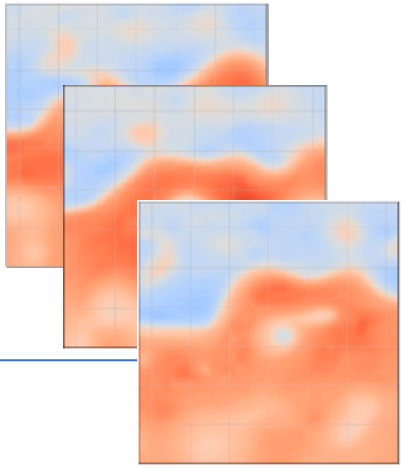
Parameterization using (learnable) ODE operator

$$\frac{\partial x(t)}{\partial t} = \mathcal{M}_\theta\left(x(t)\right)$$



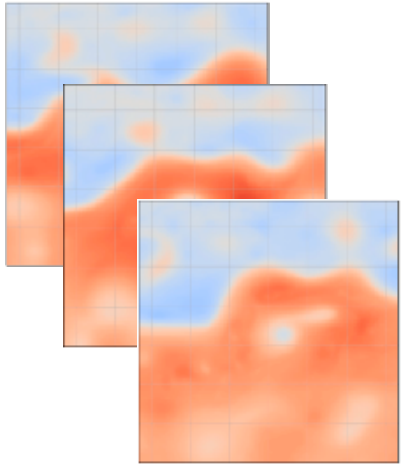
Neural integrator

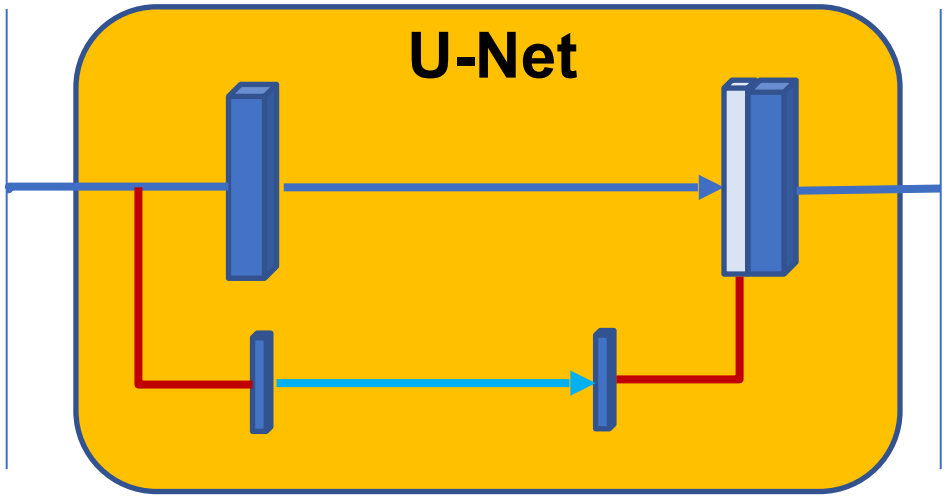
$$x(t-\Delta) + \int_{t-\Delta}^{t} \mathcal{M}_\theta\left(x(u)\right) du$$

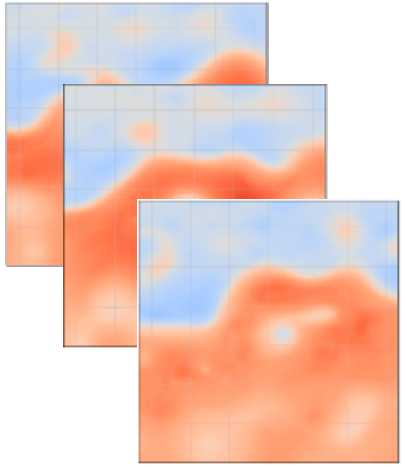


x $\Phi(x)$

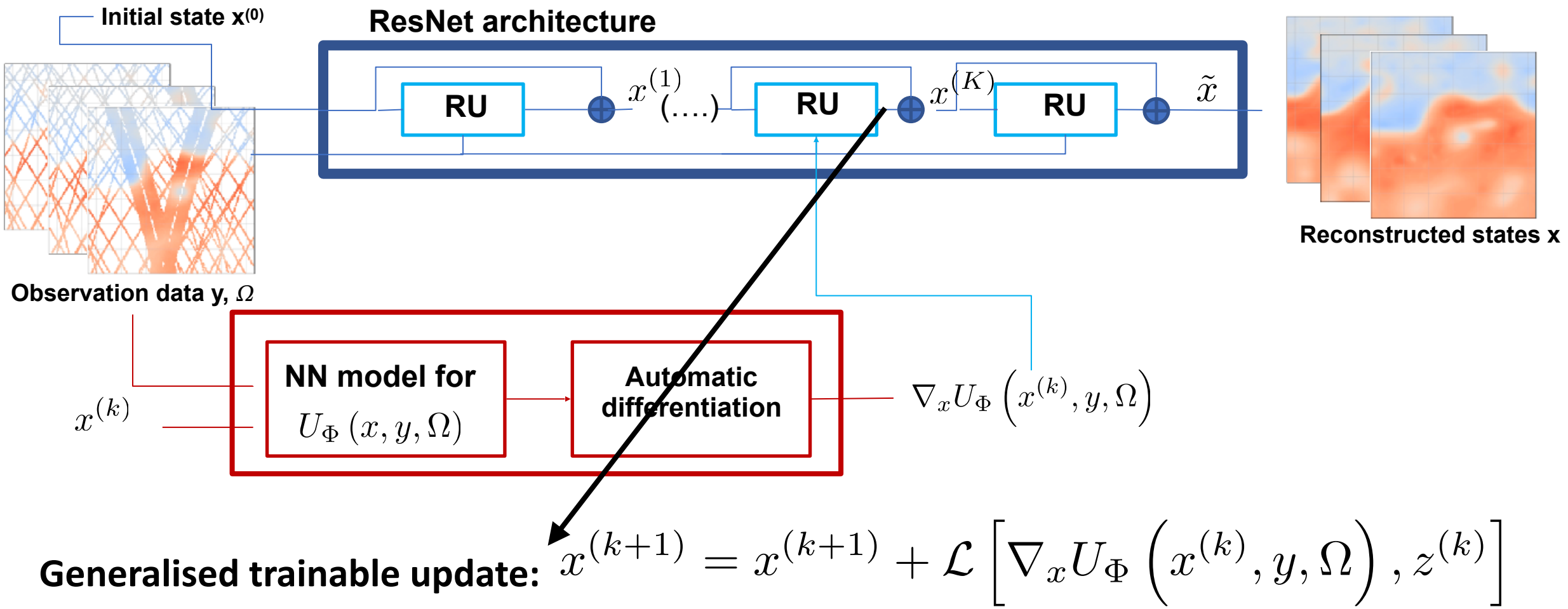
Two-scale U-Net-like Parameterization (Gibbs Field)







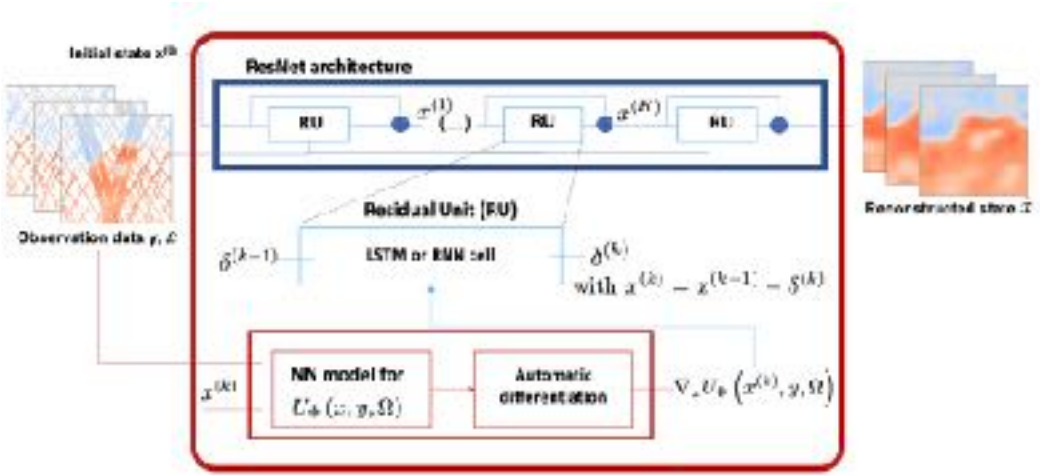
Meta-learning: optimiser learning



Generalised trainable update: $x^{(k+1)} = x^{(k+1)} + \mathcal{L}\left[\nabla_x U_\Phi\left(x^{(k)}, y, \Omega\right), z^{(k)}\right]$

Colab notebook: https://github.com/CIA-Oceanix/DLGD2021/edit/main/lecture-5-inverse-problems/notebookPyTorch_InvProb_LearningBased_4DVarNet_L63.ipynb

Model-based vs. Learning-based 4DVar DA: Unsupervised vs. Supervised scheme



Which training loss for 4DVarNet scheme ?

Unsupervised loss

$$\mathcal{L}(x, y) = \|x - y\|^2 + \lambda \|x - \Phi(x)\|^2$$

Supervised loss

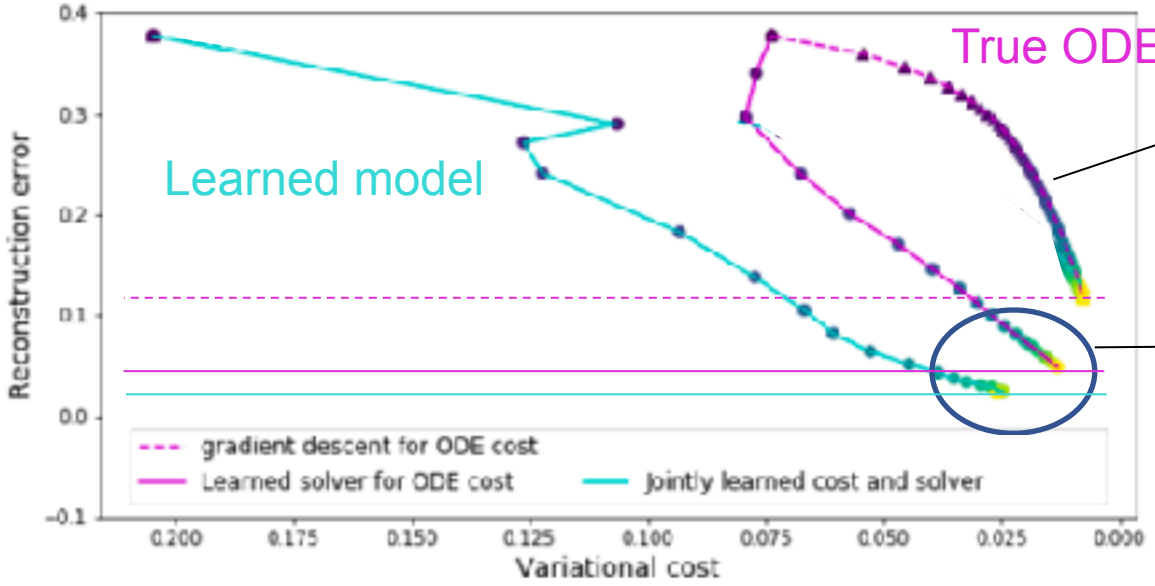
$$\mathcal{L}\left(x, x^{true}\right) = \left\|x - x^{true}\right\|^2$$

Regularisation loss

$$\mathcal{L}_{Reg}\left(x, x^{true}\right) = \|x - \Phi(x)\|^2 + \left\|x^{true} - \Phi(x^{true})\right\|^2$$

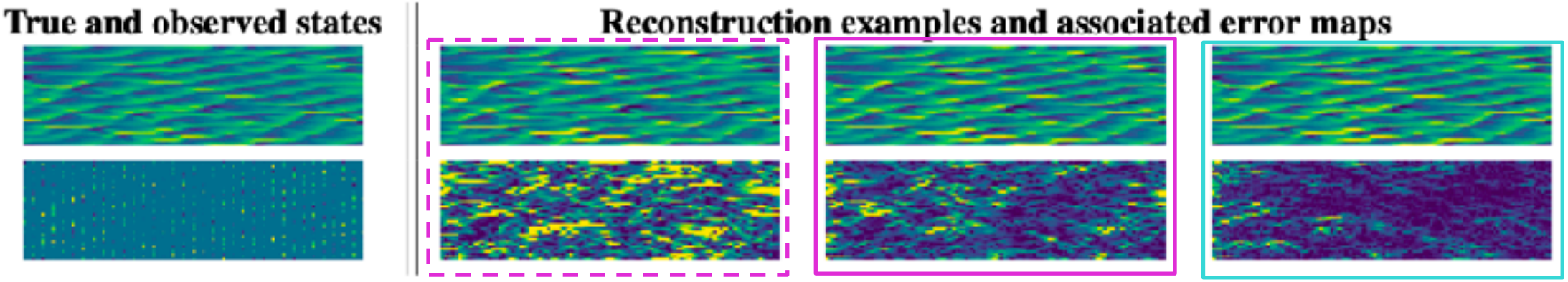
End-to-end learning for inverse problems (Fablet etal., 2020)

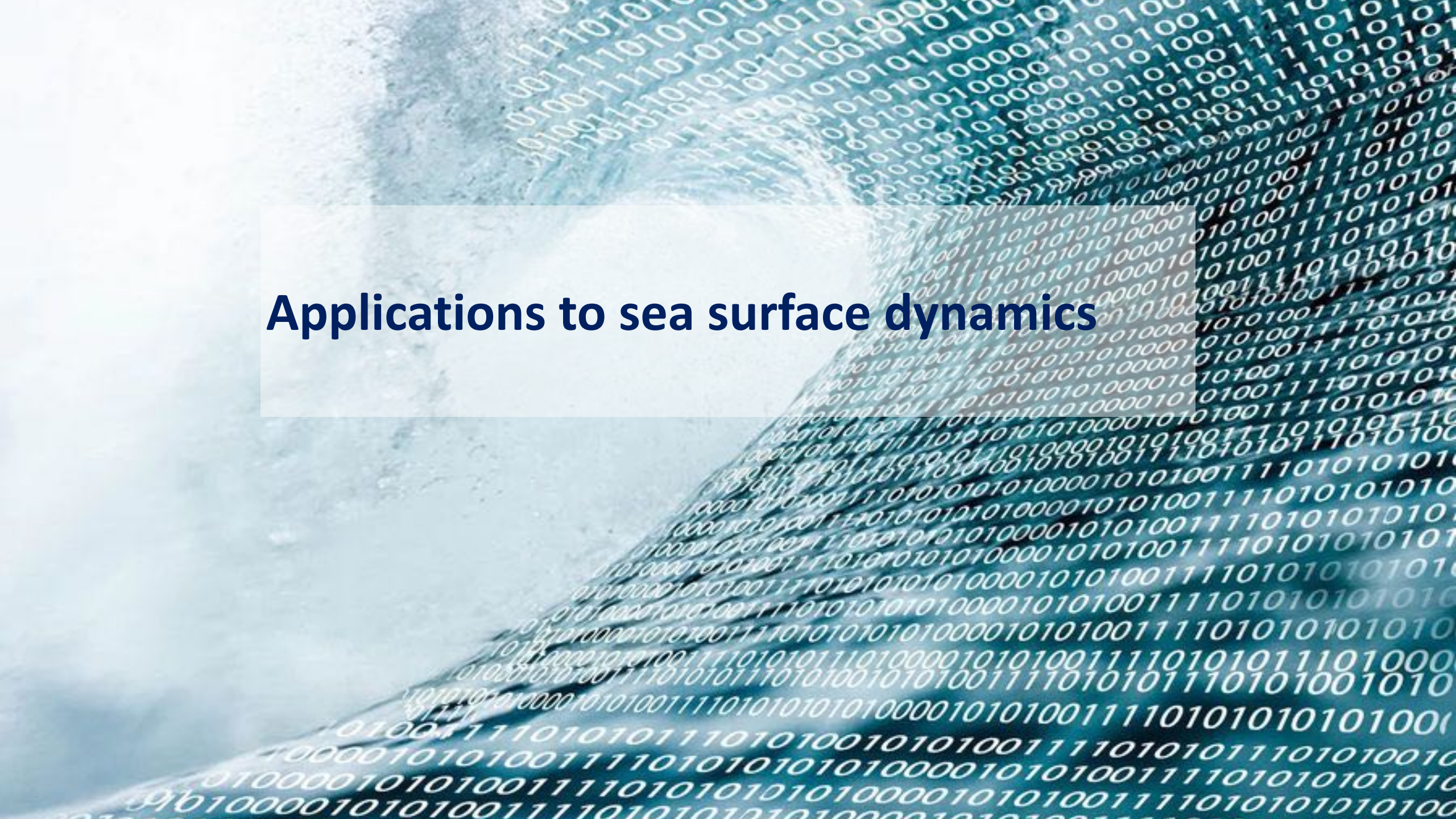
Illustration on Lorenz-96 dynamics (Bilinear ODE)



Non-supervised setting
(classic variational minimisation)

Supervised settings

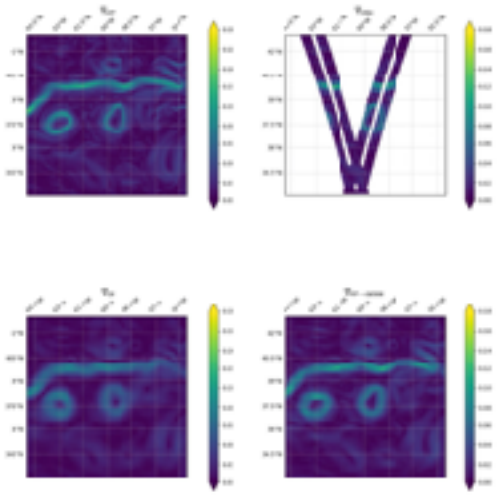


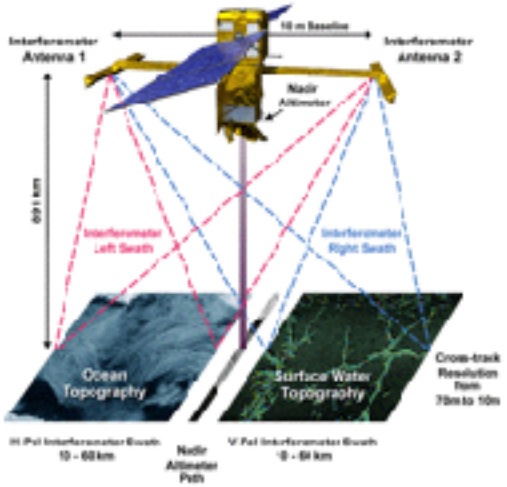


Applications to sea surface dynamics

Space-time interpolation of sea surface geophysical fields

Satellite altimetry





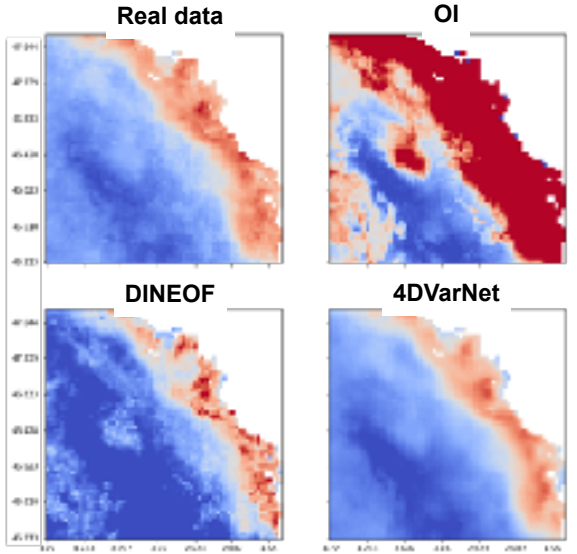
Best score for BOOST-SWOT SLA Data Challenge

duacs: 1 swot + 4 nadirs	0.82	0.72	1.22	11.16	Covariances DUACS	eval_duacs.ipynb
bfn: 1 swot + 4 nadirs	0.83	0.72	0.8	10.89	QG Nudging	eval_bfn.ipynb
dymos: 1 swot + 4 nadirs	0.83	0.72	1.2	12.07	Dynamic mapping	eval_dymos.ipynb
miost: 1 swot + 4 nadirs	0.84	0.01	1.18	10.14	Multiscale mapping	eval_miost.ipynb
4DVarNet: 1 swot + 4 nadirs	0.86	0.01	0.82	6.57	4DVarNet mapping	eval_4dvarnet.ipynb

https://github.com/ocean-data-challenges/2020a_SSH_mapping_NATL60

Sea surface suspended sediments

Metric	Dataset	Unit	Samp. Strat	OI	DinEOF	4DVarNet
RMSE	OSSE	$\log_{10}$[g/L]/m	-	0.176	0.167	**0.104**
	MODIS	$\log_{10}$[g/L]/m	Random	0.304	0.237	**0.156**
	MODIS	$\log_{10}$[g/L]/m	Patch	0.346	0.253	**0.168**
R-score	OSSE	%	-	90.4	91.3	**96.6**
	MODIS	%	Random	60.5	76.4	**89.5**
	MODIS	%	Patch	56.5	73.8	**87.3**



Mean gradient norm

Simulation and real data

Learning from real gappy data only

Collab. IMT Atl./SHOM/ LGO

Thank you.

AI Chair OceaniX 2020-2024

Physics-informed AI for Observation-Driven Ocean AnalytiX

PI: R. Fablet, Prof. IMT Atlantique, Brest

Web: https://cia-oceanix.github.io/

References

- **Meta-learning:**
  - Andrychowicz, et al (2016). Learning to learn by gradient descent by gradient descent. NIPS, 2016
  - Hospedales et al. (2020). Meta-learning in neural networks: A survey. arXiv preprint arXiv:2004.05439.
- **End-to-end learning for Inverse Problems:**
  - Dieleman et al. End-to-end learning for music audio. IEEE ICASSP 2014. doi: 10.1109/ICASSP.2014.6854950
  - Lucas et al. (2018). Using Deep Neural Networks for Inverse Problems in Imaging: Beyond Analytical Methods. IEEE SPM, 35(1), 20–36. doi: 10.1109/MSP.2017.2760358
- **Inverse problems with Plug-and-play priors:**
  - McCann et al. (2017). Convolutional Neural Networks for Inverse Problems in Imaging: A Review. IEEE SPM, 34(6), 85–95. doi: 10.1109/MSP.2017.2739299
  - Zhang et al. (2020). Plug-and-Play Image Restoration with Deep Denoiser. https://arxiv.org/abs/2008.13751
- **Deep unfolding methods:**
  - Chen et al. (2015). On Learning Optimized Reaction Diffusion Processes for Effective Image Restoration. In Proc. IEEE CVPR (pp. 5261–5269).
  - Yan et al. Deep ADMM-Net for Compressive Sensing MRI. NIPS, 2016.
  - Boudier et al. DAN -- An optimal Data Assimilation framework based on machine learning Recurrent Networks. Preprint, 2020.
  - Fablet et al. Learning Variational Data Assimilation Models and Solvers. JAMES, 2021.